National Defence / Défense nationale

A SOPHISTICATED CAD TOOL FOR THE CREATION OF COMPLEX MODELS FOR ELECTROMAGNETIC INTERACTION ANALYSIS (U)

by

Marc Dion, Satish Kashyap and Aloisius Louie
Nuclear Effects Section
Electronics Division

DEFENCE RESEARCH ESTABLISHMENT OTTAWA
TECHNICAL NOTE 91-16

PCN
041LT

June 1991
Ottawa

ABSTRACT

This report describes the essential features of the MS-DOS version of DIDEC_DREO, an interactive program for creating wire grid, surface patch, and cell models of complex structures for electromagnetic interaction analysis. It uses the device-independent graphics library DIGRAF and the graphics kernel system HALO, and can be executed on systems with various graphics devices.

Complicated structures can be created by direct alphanumeric keyboard entry, digitization of blueprints, conversion from existing geometric structure files, and merging of simple geometric shapes. A completed DIDEC geometric file may then be converted to the format required for input to a variety of time domain and frequency domain electromagnetic interaction codes.

This report gives a detailed description of the program DIDEC_DREO, its installation, and its theoretical background. Each available interactive command is described. The associated program HEDRON which generates simple geometric shapes, and other programs that extract the current amplitude data from electromagnetic interaction code outputs, are also discussed.

RÉSUMÉ

Ce rapport décrit le programme DIDEC_DREO, conçu pour les ordinateurs personnels compatibles IBM. Il utilise les librairies graphiques DIGRAF et HALO, ce qui lui permet d'utiliser diverses interfaces graphiques.

DIDEC_DREO est un programme interactif permettant la création et l'édition de structures géométriques complexes pouvant être utilisées pour l'analyse d'interaction électromagnétiques. Les structures peuvent être définies directement du clavier, par lecture de plans à l'aide d'une tablette graphique, par conversion de fichiers existant ou par génération automatique de formes simples. Les structures peuvent être converties en fichiers d'entrée pour plusieurs programmes de simulation d'interaction électromagnétiques.

Ce rapport donne une description détaillée de DIDEC_DREO, de son installation et de son utilisation, ainsi qu'un description de chaque commande. Le programme HEDRON permettant la génération de formes simples y est aussi décrit, de même que les programmes permettant d'extraire et d'inclure les résultats des programmes de simulation pour en permettre l'affichage.

EXECUTIVE SUMMARY

DIDEC_DREO is an interactive program for Personal Computers for creating models of complex geometric structures, for use in electromagnetic interaction analysis. It uses the device-independent graphics library DIGRAF and the graphics kernel system HALO, and can be executed on systems with various graphics devices.

Complicated structures can be created by direct alphanumeric keyboard entry, digitization of blueprints, conversion from existing geometric structure files, and merging of simple geometric shapes. A completed DIDEC geometric file may then be converted to the format required for input to a variety of time domain and frequency domain electromagnetic interaction codes.

This report gives a detailed description of the program DIDEC_DREO, its installation, and its theoretical background. Each available interactive command is described.

The associated program HEDRON which generates simple geometric shapes, and other programs that extract current amplitude data from electromagnetic interaction code outputs, are also discussed.

TABLE OF CONTENTS

PAGE

ABSTRACT iii

EXECUTIVE SUMMARY v

TABLE OF CONTENTS vii

1.0 INTRODUCTION 1
- 1.1 DIDEC OVERVIEW 1
- 1.2 DIDEC INPUTS 3
- 1.3 OTHER FEATURES 3

2.0 GETTING STARTED 9
- 2.1 INSTALLATION 9
- 2.2 CONFIGURING DIDEC 9
- 2.3 RUNNING DIDEC 11
- 2.4 PRINTING GRAPHICS 11

3.0 THE DIDEC PROGRAM 13
- 3.1 MODELLING STRUCTURES 13
  - 3.1.1 WIRE GRID MODELLING 13
  - 3.1.2 PATCH MODELLING 14
  - 3.1.3 CELL MODELLING 15
- 3.2 DISPLAYS 15
  - 3.2.1 DISPLAY SCALING FOR DIGITIZATION 15
- 3.3 COMMAND SYNTAX 16
  - 3.3.1 NUMERICAL EXPRESSIONS 16
  - 3.3.2 VERTEX COORDINATE SPECIFICATION 17
- 3.4 COMMAND SUMMARY 18

4.0 COMMANDS DESCRIPTION 20
- 4.1 GLOBAL COMMANDS 20
  - 4.1.1 ABORT 20
  - 4.1.2 ACTIVE 20
  - 4.1.3 DEBUG 20
  - 4.1.4 HELP 20
  - 4.1.5 HOST 21
  - 4.1.6 PRINT 21
  - 4.1.7 QUIT 21
  - 4.1.8 ROUTE 21
  - 4.1.9 STOP 21
- 4.2 DATA FILE HANDLING COMMANDS 21
  - 4.2.1 CLOSE 21
  - 4.2.2 EDIT 22
  - 4.2.3 INPUT 22
  - 4.2.4 LIST 22
  - 4.2.5 OUTPUT 22
  - 4.2.6 PURGE 22

4.2.6 ... 22
4.2.7 RENAME ... 22
4.2.8 TITLE ... 23
4.3 GRAPHIC DISPLAY MANIPULATION COMMANDS ... 23
4.3.1 BLOW ... 23
4.3.2 CODE ... 23
4.3.3 CYLINDER ... 23
4.3.4 DISPLAY ... 23
4.3.5 DOTS ... 24
4.3.6 FIELD ... 24
4.3.7 IMAGE ... 24
4.3.8 LABEL ... 24
4.3.9 LOCATE ... 24
4.3.10 NUMBER ... 25
4.3.11 RESET ... 25
4.3.12 SPECTRUM ... 25
4.3.13 TABLE ... 26
4.3.14 TEXT ... 26
4.3.15 VIEWPORT ... 26
4.4 MODEL BUILDING AND EDITING COMMANDS ... 26
4.4.1 DIGITIZATION ... 27
4.4.1.1 SCALE ... 28
4.4.1.2 DIGITIZE ... 28
4.4.1.3 CHANGE ... 28
4.4.2 VERTEX MANIPULATION ... 28
4.4.2.1 ENTER ... 28
4.4.2.2 DELETE ... 28
4.4.2.3 ADD ... 28
4.4.2.4 DISCONNECT ... 28
4.4.2.5 MULTIPLY ... 29
4.4.2.6 SOCK ... 29
4.4.2.7 VERIFY ... 29
4.4.3 WIRE MANIPULATION ... 29
4.4.3.1 BISECT ... 29
4.4.3.2 CHAIN ... 29
4.4.3.3 COLOR ... 30
4.4.3.4 CUT ... 30
4.4.3.5 DIAMETER ... 30
4.4.3.6 EDGES ... 30
4.4.3.7 JOIN ... 30
4.4.3.8 LENGTH ... 31
4.4.3.9 POLYGON ... 31
4.4.3.10 RADIUS ... 31
4.4.3.11 SEGLEN ... 31
4.4.3.12 TAG ... 31
4.4.3.13 TRIANG ... 31
4.4.4 MISCELLANEOUS ... 31
4.4.4.1 PATCH ... 31
4.4.4.2 MERGE ... 32
4.4.4.3 REFLECT ... 32
4.5 DATA FILE CONVERSION COMMANDS ... 32
4.5.1 FROM ... 32
4.5.1.1 FROM CDDP ... 33
4.5.1.2 FROM EFIE ...

4.5.1.3 FROM FDTD 33
4.5.1.4 FROM NEC 33
4.5.1.5 FROM THREDE 34
4.5.1.6 FROM TWTDA 34
4.5.2 TO 34
4.5.2.1 TO EFIE 34
4.5.2.2 TO FDTD 35
4.5.2.3 TO NEC 35
4.5.2.4 TO THREDE 35
4.5.2.5 TO TWTDA 36

5.0 ASSOCIATED PROGRAMS 37
5.1 HEDRON 37
5.1.1 BOX 39
5.1.2 CYLINDER 39
5.1.3 CONE 40
5.1.4 SPHERE 40
5.1.5 RECTANGULAR PLATE 42
5.1.6 PRISM 42
5.1.7 POLYGON 43
5.1.8 TORUS 43
5.1.9 PARABOLOID 43
5.2 ??CDDP 43

INDEX 47

REFERENCES 49

1.0 INTRODUCTION

DIDEC is an interactive program for creating wire grid, patch, and cell models of complex geometric structures, for use in the solution of electromagnetic interaction problems. Several codes are available for computation of the interaction of electromagnetic waves with simple and complicated structures. These codes exist for both frequency and time domains. The list includes NEC (Numerical Electromagnetic Code) [1] and EFIE (Electric Field Integral Equation) [2] for the frequency domain. For the time domain there are TWTD (Thin Wire Time Domain) [3], FDTD (Finite Difference Time Domain) [4], and THREDE [7] among others. These codes have been extremely useful in finding numerical solutions to many electromagnetic interaction problems.

All these codes have one feature in common: they require as an input the geometry of the object under study. The geometry is supplied as a wire grid, surface-patch, or cell model, depending on the simulation code. The creation of such geometric data is not an easy task, and the difficulty increases with the complexity of the structure (e.g., an aircraft). To help in this regard, DIDEC (Digitize, Display, Edit and Convert) [5] was developed at Concordia University a few years ago. Recently, DREO has made several improvements to DIDEC.

DIDEC began as a program that created wire models of a geometric structure using a digitizing tablet. Initially, it created NEC input files for wire grid structures. Since then it has developed into an interactive program, DIDEC_DREO, for designing wire grid, patch, and cell models of complex structures for solving electromagnetic interaction problems. This report describes various features of the present version of DIDEC_DREO.

1.1 DIDEC OVERVIEW

Figure 1-1 shows a block diagram of the present DIDEC_DREO structure. A geometric structure is stored in a DIDEC data file (*.DID). The wire grid portion is stored in vertex-edge form and the patch portion is stored in vertex-patch form. In other words, a *.DID file contains a list of vertex numbers and their (cartesian) coordinates

$$V_1 = (X_1 , Y_1 , Z_1)$$
$$\vdots$$
$$V_n = (X_n , Y_n , Z_n)$$

followed by a list of edges defined in terms of the two end vertices

$$\vdots$$
$$E_k = (V_i , V_j)$$
$$\vdots$$

and a list of patches defined in terms of their corners

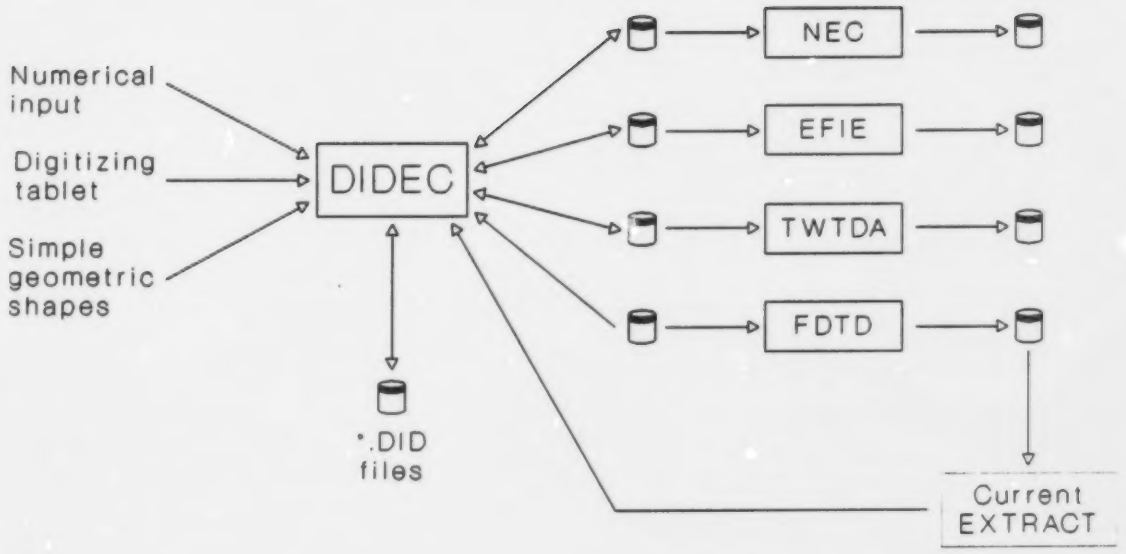

Figure 1-1 Block diagram showing the structure of the DIDEC program.

$$P_t = \langle V_i , V_j , V_k \rangle \qquad \textit{for a triangular patch}$$

$$P_q = (V_i , V_j , V_k , V_\ell) \qquad \text{for a quadrilateral patch}$$

Additional information in *.DID includes wire or patch colour which can represent wire diameter, wire length, patch area or any electromagnetic characteristic such as current magnitude.

1.2 DIDEC INPUTS

DIDEC_DREO is a very powerful tool, mainly due to its highly interactive capabilities. DIDEC_DREO allows structures to be created or altered interactively. Several files can be merged together, allowing the user to define complicated structures "by parts". The results associated with one or multiple input or output files can be displayed dynamically from various projections or viewpoints simultaneously on one or more viewports. Figure 1-2 shows an example of a typical 4-viewport display.

Structures can be created in different ways:

- Vertices can be entered numerically from the keyboard and then be joined to form wires or patches.
- A graphic tablet can be used to digitize a set of blueprints. Each vertex may be digitize twice with different view to resolve all coordinates, or cross-section planes may be specified. Very large blueprints can be digitized in sections.
- A geometric structure can be read from the input file of one of the electromagnetic analysis programs (NEC, EFIE, TWTD, FDTD, THREDE, etc.).
- Simple geometric shapes, such as box, cylinder, cone, sphere, plate, prism, polygon, torus or paraboloid, can be created with a companion program and then merged to form complex structures.

Various manipulations of data in *.DID are possible. Vertices, wires and patches can be added, moved, removed, altered, etc., and several *.DID files can be merged into one.

1.3 OTHER FEATURES

When a *.DID file is completed, the geometric structure can be readily converted to a file in the format needed for a specific electromagnetic analysis program (NEC, EFIE, TWTD, FDTD, THREDE, etc.). It may however be necessary to adapt the model before to meet some specific requirements of a particular code.

Structures can be defined in terms of wires (wire grid models) or surfaces (patch models). An example of each is shown on Figure 1-2 (top left and top right respectively). DIDEC allows wire grid models to be converted to patch

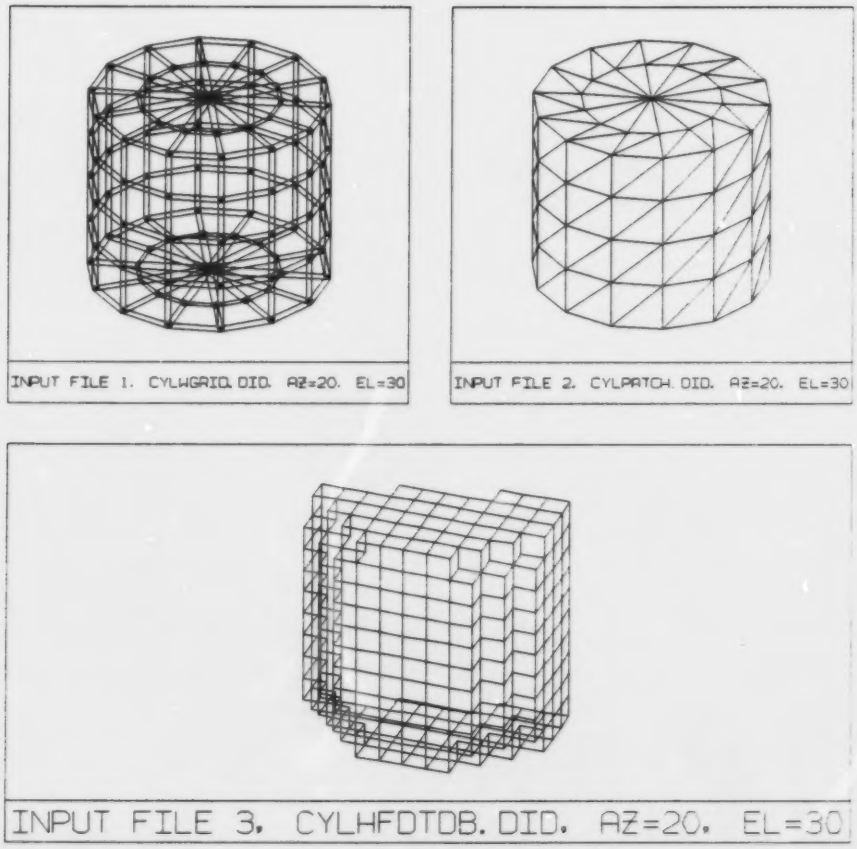

Figure 1-2 Various representations of a cylinder: wire-grid modelling (top left), patch modelling (top right), and cell modelling (bottom).

models, and vice versa. The conversion from wire grid to patch structure in DIDEC uses the mathematical theory of graphs. The "incidence matrix" for the wire grid structure is analyzed to identify all triangular and quadrilateral wire frames, which can be converted into triangular and quadrilateral patches. Each quadrilateral wire frame can also be divided into two triangular wire frames first, and then the latter converted into triangular patches.

There are additional considerations when converting from a wire grid structure to a patch structure. Some codes, such as EFIE, have a geometric restriction that an edge cannot be shared by more than two triangular patches. For example, when converting the NEC wire grid model of the aircraft CP140 to EFIE (Figure 1-2, Figure 1-3, top and bottom respectively), special care has to be taken where the wings and the tail fins join the fuselage. The solution is to "inflate" the wings and fins to thin tubes (i.e. "socks"). In DIDEC_DREO this process is automated: a *.DID file is analyzed to identify all the edges that are shared by more than two patches, and then in each case the extra "wings" are chosen for "inflation". The right wing of the CP140 aircraft, before and after inflation, is shown on Figure 1-4.

Some electromagnetic interaction codes, such as FDTD and THREDE, use a cell model to represent structures. In cell modelling, the geometric structure is represented using orthogonal cellular blocks (cells). An example is shown on Figure 1-2 (bottom). With the current version of DIDEC_DREO, it is possible to read cell model files (input files to FDTD or THREDE) for display purposes only. It is, however, possible to convert any DIDEC file (ie. wire grid or patch models) to a FDTD or THREDE input file, suitable for use in running the finite-difference time-domain simulations codes. Those files created by DIDEC may then be reconverted back to a DIDEC format for display. Figure 1-5 shows a EFIE patch model (in this case, the walls of the bridge of a Canadian Patrol Frigate) and its conversion into a FDTD input file.

DIDEC_DREO also has the capability to read, after an electromagnetic interaction program has been run, the results such as the current magnitude data, which can be added to a *.DID file, so that they can be displayed by DIDEC as a colour code for each wire or edge.

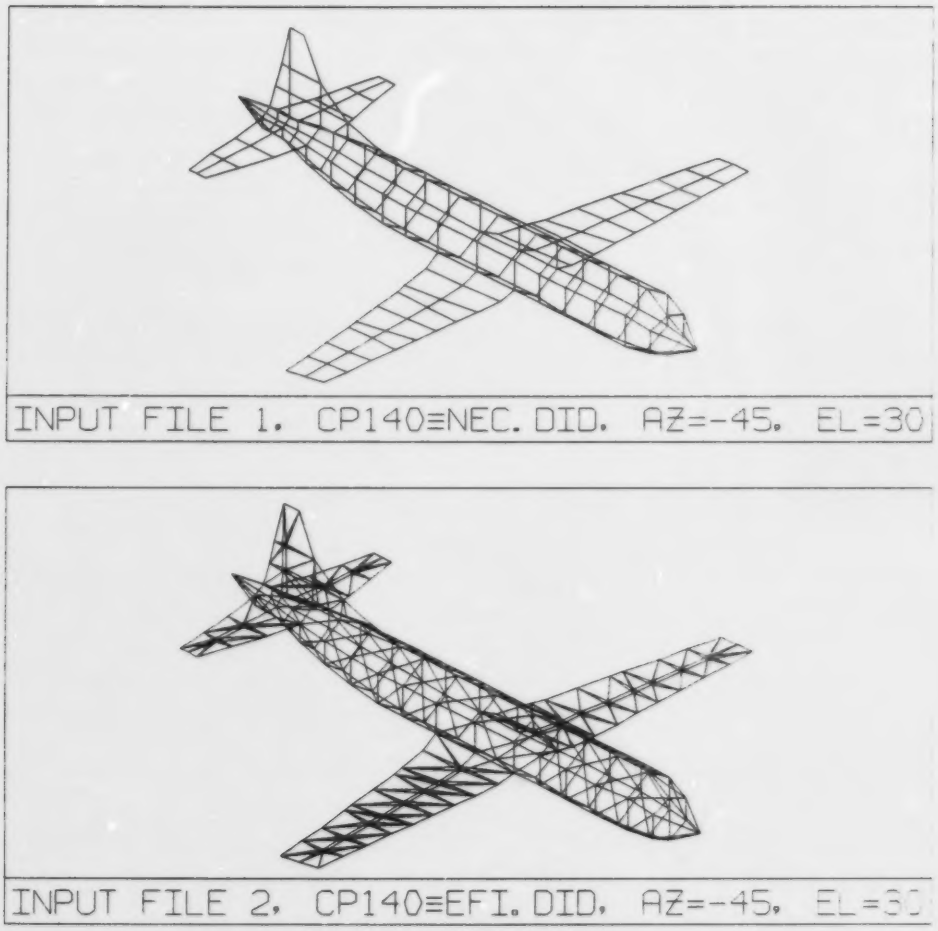

Figure 1-3 Wire grid and patch models of the CP140 aircraft.

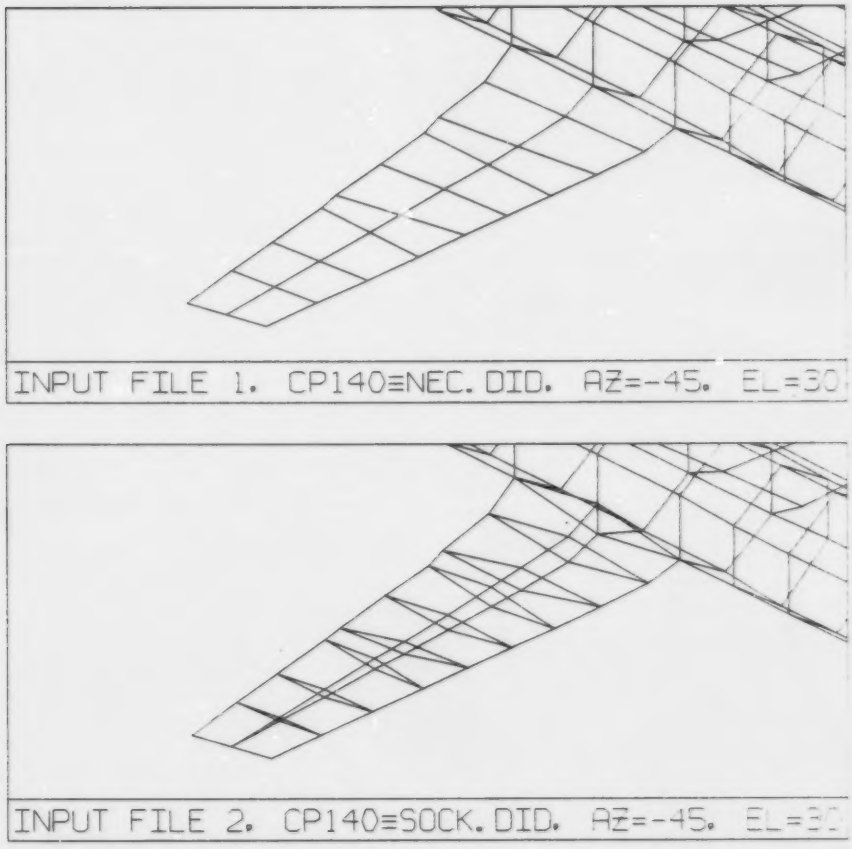

Figure 1-4 Right wing of the CP140 aircraft, before (top) and after (bottom) inflation.

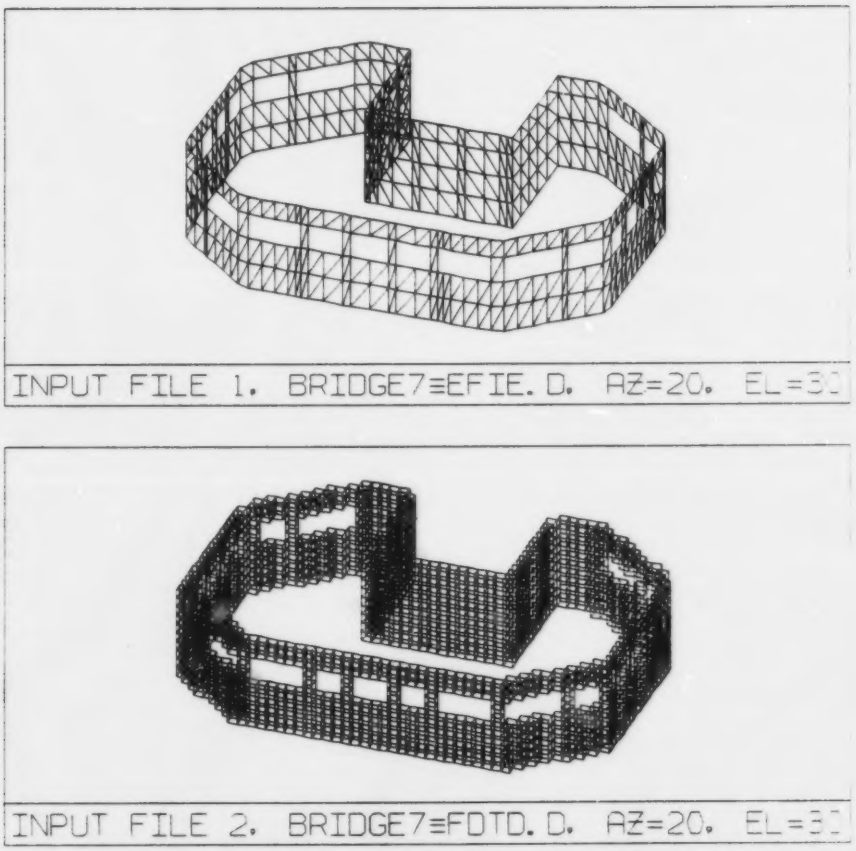

Figure 1-5 Patch model (top) and cell model (bottom) of the bridge of the Canadian Patrol Frigate.

2.0 GETTING STARTED

This chapter gives the installation and operation procedures for the DIDEC_DREO program.

The current version is DIDEC_DREO, July 1990 (Copyright 1990, DREO), is written in Microsoft FORTRAN 4.1, and runs on an IBM-compatible PC under DOS 3.30 or DOS V4. It uses the device-independent graphics package DIGRAF (Copyright 1979, University of Colorado), and the graphics kernel system HALO 3.0 (Copyright 1989, Media Cybernetics).

2.1 INSTALLATION

The DIDEC_DREO program is distributed on a single floppy disk which contains two directories:

\DIDEC	This directory contains all the executable files, including overlays, configurations and device drivers files.
\DIDEC\HELPFILE	This subdirectory contains all the help files used by the DIDEC HELP command.

The installation is simply done by copying all the files to a hard disk. The following DOS command transfers all files to the directory \DIDEC onto a hard disk:

```
XCOPY [floppy_drive]:\ [hard_drive]:\ /S
```

2.2 CONFIGURING DIDEC

DIDEC uses the HALO device-independent graphics kernel. HALO is a collection of Fortran-callable subroutines for developing PC-based graphical applications. HALO routines are device-independent and a wide variety of device drivers is supplied for graphic displays, locators and printer/plotters. Standard graphic modes (EGA, VGA, etc.) as well as higher resolution modes (super VGA) are supported for many popular graphic adapters.

The HALO configuration file HALO.CNF must be present in the \DIDEC directory. This configuration file is set to reflect the system's hardware. It lists the drivers used by HALO and their mode. All the files listed must reside in the \DIDEC directory. The HALO.CNF file has the following format:

```
-D[graphics_device_driver]
-M[mode_number]
-P[printer_driver]
-L[locator_device_index&interrupt]
-T[COM_port&digitizer_format_file]
```

The first two lines specify the display device driver and the mode of operation. A description of the drivers available to HALO can be found in Section 2 of the HALO Language and Device Reference manual. A combination that gives the highest resolution with 16 colors should be used.

The third line specifies the device driver for the printer. "-P[blank]" denotes the absence of printer (hence the DIDEC "PRINT" command cannot be used). A complete list of the printer drivers can be found in Section 3 of the HALO manual.

For example, the HALO.CNF file:

```
-DHALOIBMV.DEV
-M7
-PHALOLJTP.PRN
-L25
-T2CALCOMP.DGT
```

selects the drivers for a standard VGA adapter (mode 7) and for a HP LaserJet printer.

The fourth line is used to configure the locator device (such as a mouse) used by some DIDEC commands. The first digit is the locator device index: "-L0" if the locator is absent, "-L1" for a HALO-supplied locator device driver, and "-L2" for a manufacturer-supplied driver. See Section 5 of the HALO manual and the description of the HALO command "SETLOCATOR". The second digit specifies the interrupt level used by the locator. In the example above ("-L25"), a Microsoft mouse is attached to interrupt 5. Locator device drivers are DOS-resident programs and so it is necessary to load the locator driver before running DIDEC. If there is no locator device, some DIDEC commands that normally use it can still be used via direct keyboard input of parameters, and others cannot be used. The DIDEC "ACTIVE" command toggles the locator availability indicator if a locator exists.

DIDEC allows the input of geometric structures through a digitizer over blueprints. This requires a digitizer attached to a COM port. The last line of HALO.CNF is used to configure the digitizer: "-T0" indicates there is no digitizer (hence the digitizing DIDEC commands "SCALE", "DIGITI", and "CHANGE" cannot be used), "-T1" and "-T2" indicate the digitizer is connected to COM1 and COM2, respectively. If a digitizer is specified, the name of a digitizer format file must follow and this file must be located in the directory \DIDEC. This file contains three lines:

```
Left, right, top, & bottom limits of tablet     (4 integers)
Format of digitizer input                  (character string)
Translation table of keypad cursor           (13 characters)
```

For example, if the digitizer is a CalComp 9000 tablet with a 16-key cursor in the point mode, the format file CALCOMP.DGT would be:

```
0 47999 0 35999
(A1, 1X, 2I5)
0123456789DCA
```

The first line specifies the range of the digitizing tablet for both *x* and *y* coordinates. The second line is a FORTRAN format, enclosed in parentheses, required to decode the input of the table when a point is digitized. Three variables are decoded: the ASCII value of the key pressed on the keypad and the integer giving the cursor location (*x* and *y* coordinates). Only these three values should be decoded and all other characters received the digitizer input should be skipped (by the "X" format descriptor). The value for the key pressed can be received either before or after the *x* and *y* coordinates. Therefore, the two possible formats are: "(A1,2In)" and "(2In,A1)", with possible "nX" inserted in it. The third line gives 13 different characters sent by the digitizer that would be, in order, interpreted as the numbers 0-9, "done", "erase", and "abort". The keypad cursor used must therefore be able to send 13 or more different signals.

2.3 RUNNING DIDEC

DIDEC also requires the ANSI.SYS driver to execute. The following line should be present in the CONFIG.SYS file:

```
DEVICE=C:\dospath\ANSI.SYS
```

The current version of DIDEC.EXE uses overlays and should be executable even when several DOS-resident programs are loaded. If it becomes necessary, a memory manager such as QEMM may be used to move some drivers to high memory, thus saving more space for running DIDEC.

To run DIDEC, first move to the working directory \DIDEC, then execute by entering "DIDEC" at the DOS prompt. Alternatively, DIDEC can be executed from any directory, providing that the following line is executed before or is placed in the AUTOEXEC.BAT file:

```
APPEND [hard_drive]:\DIDEC
```

DIDEC can then be started with [hard_drive]:\DIDEC\DIDEC, via a batch file or its location can be specified with the DOS PATH command.

DIDEC is an interactive program and reads in a sequence of user-supplied commands at the DIDEC prompt "->". The "HELP" command may be used to display a list of the available commands or to give more detailed explanations of any one specific DIDEC command. Most DIDEC commands prompts for additional user-input. A command in progress can be aborted by entering "ABORT" at these additional prompts, in which case the program returns to the DIDEC prompt "->". To terminate DIDEC, enter the command "QUIT" or "STOP".

2.4 PRINTING GRAPHICS

Any displayed DIDEC graphics image can be printed by entering the "PRINT" command (if a printer is attached and is specified on the "-P" line in HALO.CNF). The printed image is also saved in the file HALO.PIC. One can also use the "IMAGE" DIDEC command to save the graphics image to a specific *.PIC file for later viewing and printing.

To recover a *.PIC graphics image file to the screen, execute the HALOREAD program:

```
HALOREAD file_name                    (without the PIC extension)
```

The saved image will be displayed on the screen and the user is prompted for printing. A *.PIC file can be moved to other PC's for recovery and printing by HALOREAD, but the graphics device driver and mode must be the same as they were in the original PC where the file was created.

3.0 THE DIDEC PROGRAM

The name "DIDEC" is derived from Digitize, Display, Edit and Convert program. It is a Computer Aid Design tool for building wire grid, patch and cell models of structures for use with several electromagnetic interaction codes. This chapter describes the general command syntax and gives a list of the commands.

3.1 MODELLING STRUCTURES

The geometric input for the electromagnetic interactions codes can be classified into three general types of models: wire grid modelling (NEC and TWTD), patch modelling (NEC and EFIE), and cell modelling (FDTD and THREDE). With wire grid modelling, a structure is represented as a set of wires (thin or fat) connected at their end points. Surfaces are approximated by using meshes. With patch modelling, a structure is represented as a set of patches connected at their edges. Codes may allow open or close bodies. In cell modelling, a structure is considered to be occupying a collection of cartesian grided mesh locations ("cells") in space. The following sections discuss each type of modelling in more detail.

3.1.1 WIRE GRID MODELLING

Complex structures can be modelled by using conducting wires, with each wire possibly subdivided into a number of segments. A wire grid mesh can model accurately a conducting sheet, providing that the mesh spacing is small compared with the wavelength. For instance, guidelines for NEC suggests that the segment length should be less than $\lambda/10$. Wires can be thin or fat. Wires are considered thin when their radius is very small compared to the wavelength. Some codes, such as TWTD, support only thin wires, while others will support thin and fat wires. For thin wires, the current is approximated as a filament of current on the segment axis, and for fat wires, the current is distributed uniformly on the segment surface.

Wires are specified in terms of their endpoint vertices, i.e. their 2 VIDs. The wire characteristics supported are diameter, number of subdivisions, and colour. The diameter of each wire is classified by a user-defined set of up to 15 different diameters. The number of subdivisions is implemented as a maximum length per subdivision, also classified by a user-defined set of 15, and converted automatically to the appropriate number of subdivisions when the "TO" command is used. Wire colour is used only for display purposes. The colour parameter may thus be used to identify particular groups of wires. Displays normally show the wire colour, but may instead show colour-coding be either diameter or maximum subdivision length.

Colour is selected using the "COLOR" command, described in Section 4.4.3. When a wire is created it is given the current modal colour, which defaults to white. The colour of existing wires may be changed by entering the colour command and typing the wire endpoints in response to the prompt. The new colour is then specified using a palette on the graphics display if the display device is interactive, or by typing the colour number if the display device is not

interactive. The current modal colour is changed by using the "COLOUR" command without specifying any wire IDs.

Diameter is specified using the "DIAMETER" and "TABLE" commands, described under 'Wire Manipulation Commands'. There may be up to 15 different diameters. Each is assigned an integer index from 1 to 15. The correspondence between these indices and the actual diameter is defined by a diameter table that is created using the "TABLE" command. When wires are created they are given the current modal diameter index, which defaults to 1. The diameter of existing wires may be changed by entering the "DIAMETER" command and typing the wire endpoints in response to the first prompt. The new diameter index of these wires is prompted. The current modal diameter index is changed by using the "DIAMETER" command without specifying any wire id's. The "TABLE" command also associates a colour with each diameter entry in the table. This colour is used for the 'Colour-Coding by Diameter' option in the "DISPLAY" command.

The maximum length per subdivision is handled similarly to the "DIAMETER" command, using another table. If no subdivisions are desired then a very large length should be specified.

3.1.2 PATCH MODELLING

Structures can also be modelled by means of multiple, small flat surface patches. Some codes, such as NEC, allow the use of wires and patches simultaneously. In that case, it is also possible to connect wires to patches. As with the wire grid models, the patches should be small compared with the wavelength; typically, a minimum of 25 patches should be used per square wavelength of surface area. Patch modelling can be used to represent close bodies or open bodies (bodies with apertures or cavities).

DIDEC supports triangular and quadrilateral patches. Patches are created by the "PATCH" command. They are stored in a *.DID file as numbered patches defined by the corner vertices. For quadrilateral patches, the four corners need not to be coplanar. The "TO NEC" command converts them to NEC patches (SP and SC cards). The "TO EFIE" command converts all patches into a list a vertices and edges to create an *.EFI file.

The EFIE triangular "patches" are actually stored in a *.DID file as triangular "wire grids" with an arbitrarily defined radius as 10% of the wire length. This is because the EFIE program treats the triangular patches simply as triangles with thin edges (i.e. line segments with zero radius). Only the information on vertices and edges are supplied in the EFIE input file *.EFI, with the triangular patches configured internally by EFIE.

EFIE also imposes some additional limitations. It supports only triangular patches. The "TRIANG" command can be used to automatically divide all quadrilateral patches into two triangular patches. EFIE also prohibits that an edge be shared be more than two patches. The "EDGES" command can be used to find any edges shared by more than two triangular faces, and the "SOCK" command can be used to inflate the "wings". See Section 4.4 for the description of the individual commands.

3.1.3 CELL MODELLING

In cell modelling, the neighbourhood of the geometric structure (the "cell space") is divided into cartesian grided mesh points (i.e. into a box-like tessellation). The structure is considered to occupy a collection of these mesh locations ("cells"). Thus depending on the size of the cells, a cell model is a "staircase" approximation of the geometric structure, when the structure has curved surfaces or planer surfaces that are not parallel to the cartesian axes.

The FDTD/THREDE geometric structure of cells is represented in DIDEC as orthogonal cellular wire grids (again with an arbitrary wire radius as 10% of wire length). The conversion commands "FROM FDTD" and "FROM THREDE" are used to display the cell structure in DIDEC only: the resulting *.DID files cannot be used to generate any geometric input file to run another electromagnetic simulation code.

It is, however, possible to convert any wire grid or patch structure in *.DID format to a cell model, with the commands "TO FDTD" and "TO THREDE", suitable for use in running FDTD and THREDE.

3.2 DISPLAYS

There may be from one to four independent viewing areas on the screen, called 'viewport'. Viewport configuration (shape and location) is limited to a given set. Each viewport (if not empty) corresponds to exactly one open data file or table, but a file may be displayed in more than one viewport. Any change to the output file will be shown immediately in all viewports associated with the output file. A viewport may be in use or empty. If empty, it may or may not be erased. When a viewport has just been defined using the "VIEWPORT" command, it is both empty and erased. If a viewport contains a display but the file from which the display was done is no longer open, then the display is empty but not erased. Certain commands which require specification of a viewport will abort if the viewport is in use but will erase it and use it if it is empty and not erased.

3.2.1 DISPLAY SCALING FOR DIGITIZATION

One viewport is selected (see below) as the 'primary' digitization viewport. It is scaled based on the 4 corners of the tablet, and the appropriate 2-D projection is used. Any other viewports associated with the output file will not be rescaled, although any digitized vertices which happen to fall within the viewports will be drawn if the defined coordinates are compatible with the 2-D projection in use. Therefore, if the user wants a non-standard view such as a 3-D view while digitizing then he must use more than one viewport and set up the others before the "DIGITIZE" command (he cannot use a 2-D projection alone, there must always be a 'primary' viewport).

The primary viewport is chosen as part of the "SCALE" command and may be specified by the user by giving the viewport number. The default is as follows:

- if no viewports are defined, then the full screen is defined as a viewport associated with the output file; and

- if the viewport configuration is defined, then the lowest-numbered free viewport is used. If none are free then the user must choose which viewport is to be erased.

3.3 COMMAND SYNTAX

This section describes the general syntax for user commands. When the "->" prompt is displayed, DIDEC is ready to accept a user command. Commands may be abbreviated as long as enough letters are entered so that the command can be uniquely identified. For example, "QUIT" may be abbreviated to as little as one letter, but "CHAIN" cannot be abbreviated to "C", "CH" or "CHA" since these do not distinguish "CHAIN" from "CHANGE". The command name must be followed by a carriage return, often shown as "<CR>".

Most commands require further keyboard input. After the command is entered, the user is prompted to enter the required data. In a few cases this data may instead be entered on the same line as the command itself, but this is never required. The term "parameter" is used to represent one data value. If more than one parameter is entered on the same line, they must be separated by commas. The following are the possible parameter types:

- Null (empty or extra comma(s))

- Integer value or expression (see Section 3.3.1 below)

- Real (floating point) value or expression

- Range of consecutive integer values, expressed as two integers separated by a hyphen (the first value must always be less than the second)

- Wire identification, expressed as two integers separated by a space

- Character string, such as a filename or title

3.3.1 NUMERICAL EXPRESSIONS

Whenever an integer or real value is expected, a numerical expression may be used instead. The five basic operations (including exponentiation) are allowed, as well as unlimited parentheses. The syntax and priority of operations are the same as FORTRAN. All input values are considered to be real (floating point) numbers, so truncation is never done. The final result is rounded, not truncated, if the prompt being answered expects and integer result. Vertex coordinate specifications (VCS, described below) may be used in expressions.

The following are valid expressions and are all equal:

```
10              14-4
5*2             3**2+1
6+(5+3)/2       6+4*2/2
```

3.3.2 VERTEX COORDINATE SPECIFICATION

A vertex coordinate specification (VCS) may be used whenever an integer or real value is expected. A VCS is a way of accessing the value of the X, Y, or Z coordinate of an existing vertex. The full form of a VCS is:

```
FnVmC
```

where n is a file number, m is a vertex number, and C is either X, Y, or Z. The VCS consists of 3 parts, Fn, Vm, and C. Any one or two of these parts may be omitted, but at least one must be present so that the VCS can be recognized. The defaults are:

Fn: File - the output file
Vm: Vertex - the most recently accessed vertex, usually clear from the context
C: X, Y, or Z - defined by the context if possible, else must be specified

The defaults for file and vertex are used if a number is not specified, whether or not the Key letter "F" or "V" is specified. The only purpose in specifying "F" for "V" without a number is due to the restriction that a VCS cannot be recognized unless one of the letters "F", "V", "X", "Y", or "Z" is specified.

The VCSs are extremely useful in specifying coordinates of vertices required for digitized segment realignment and other situations where a direct numeric coordinate entry is required or desired, since their real numeric value may then be given to the system only once (say, when digitizing the first segment containing that alignment vertex). All subsequent references can be given using a VCS, reducing thereby any consistency errors in repeatedly entering the same (possibly long) numeric values.

As an example of the uses of a VCS, consider the "ENTER" command, which allows numerical entry of vertex coordinates. Suppose that vertex 1 in the output file is defined, and vertices 2 and 3 are now ENTERed as follows:

```
-> ENTER
Vertices? 2,3
Vertex #2 X, Y, Z? V1*2
Vertex #3 X, Y, Z? V1,V2,X*2
```

All 3 coordinates of vertex 2 will then be double those of vertex 1, since the expression "V1×2" is evaluated 3 times, once for each coordinate of vertex 2, and each time the corresponding coordinate of vertex 1 is used. Vertex 3 has the X coordinate of vertex 1 and the Y coordinate of vertex 2. The Z coordinate of

vertex 3 is double the previous X coordinate, resulting in an error if vertex 3 was not previously defined. Note that the default vertex used in evaluating "X×2" was vertex 3, although vertex 2 had just been referenced, because referencing a vertex in a VCS does not change the definition of "most recently accessed vertex" for the purpose of determining the default. Also note that until the definition of vertex 3 is complete, any VCS which references vertex 3 uses the values from the previous definition, not the current one.

3.4 COMMAND SUMMARY

There are 60 available DIDEC commands, a list of which may be obtained with the "HELP" DIDEC command. More information on a specific command is obtained by typing HELP followed by the name of the command.

The commands can be divided into five groups:

a. global commands
b. data file handling commands
c. graphic display manipulation commands
d. model building and editing commands
e. data file conversion commands

An alphabetical list of all commands follows:

ABORT	abort any command in progress
ACTIVE	toggles the availability tag of a locator device
ADD	adds a constant to coordinates of chosen vertices
BISECT	bisects a wire
BLOW	zooms into a part of a display
CHAIN	creates a chain of wires by joining a list of vertices
CHANGE	re-digitizes vertices already defined
CLOSE	closes a file
CODE	selects the color-coding table to be used for displays
COLOR	changes the color of specified wires, or the color to be given to subsequently created wires
CUT	deletes wires
CYLINDER	toggles drawing of wires as cylinders
DEBUG	for use only when debugging DIDEC
DELETE	deletes vertices
DIAMETER	changes the diameter table index of specified wires, or the index to be given to subsequently created wires
DIGITIZE	digitizes vertices which are undefined or partially defined
DISCON	deletes vertices with ≥ a specified number of wires attached
DISPLAY	displays a geometry data file
DOTS	toggles display of vertices
EDGES	picks out all edges shared by more than two faces
EDIT	opens an existing file as the current output file
ENTER	accepts numerical keyboard entry of vertex coordinates
FIELD	displays incident field
FROM CDDP	creates a DIDEC file from a CDDP file and put CURRENTS to the COLOR field for CODE=C display
EFIE	creates a DIDEC file from an EFIE file

```
     FDTD    creates a DIDEC file from a FDTD geometry file
     NEC     creates a DIDEC file from a NEC file
     THREDE  creates a DIDEC file from a THREDE geometry file
     TWTDA   creates a DIDEC file from a TWTDA file
HELP         displays this list.  HELP Command_Name gives information on
             the specified command.
HOST         issues host (DOS) commands
IMAGE        writes the current display to a HALO image *.PIC file
INPUT        opens an existing file for input (read-only)
JOIN         creates wires joining existing vertices
LABEL        prints out a list of vertices at the locator cursor
LENGTH       changes the "maximum subdivision length" table index of
             specified wires, or the index to be given to subsequently
             created wires
LIST         prints a formatted listing of the contents of a file
LOCATE       labels 3D locations on displayed files
MERGE        merges an input file into the output file
MULTIPLY     multiplies coordinates of chosen vertices by a constant
NUMBER       indicates vertex and tag displayed number size
OUTPUT       creates a new output file
PATCH        creates triangular or quadrilateral patches
POLYGON      creates a chain of wires, and closes the chain
PRINT        sends displayed image to a printer
PURGE        destroys a file
QUIT         exits DIDEC even if an output file is open
RADIUS       changes the absolute radius value of a wire
REFLECT      reflects a file about a principle axis (x,y,z)
RENAME       changes the name of the output file
RESET        resets scaling information for a viewport and redraws it
ROUTE        accepts command input from a file
SCALE        sets the scale of the digitizing tablet
SEGLEN       changes the absolute segment length value of a wire
SOCK         splits vertices in two and doubles the connecting wires
SPECTRUM     displays current magnitude spectrum
STOP         exits DIDEC if no output file is open
TABLE        reads, creates and updates color-coding, diameter, length, and
             area tables
TAG          gives a wire a NEC-compatible identification tag
TEXT         writes text at cursor-chosen location
TITLE        displays (I/O) and changes (O) file titles
TO EFIE      creates an EFIE input file
   FDTD      creates a FDTD geometry file
   NEC       creates a NEC input file
   THREDE    creates a THREDE geometry file
   TWTDA     creates a TWTDA input file
TRIANG       splits each quadrilateral into two triangles
VERIFY       shows which vertices are fully or partly defined
VIEWPORT     chooses a display viewport format
```

4.0 COMMANDS DESCRIPTION

This chapter gives a detailed description of the DIDEC commands, divided into five different groups: global commands, data file handling commands, graphic display manipulation commands, model building and editing command, and data file conversion commands.

4.1 GLOBAL COMMANDS

Commands in this group provides general system-level interaction.

4.1.1 ABORT

This command is in fact not a DIDEC-level command. It is a possible response to most prompts WITHIN the progress of a DIDEC command. If "ABORT" is entered, the current command is aborted and the program returns to a DIDEC prompt ("->"), when another DIDEC command may be entered.

4.1.2 ACTIVE

This command toggles the availability tag of a locator device (joystick, trackball, mouse,...), if a locator exists (i.e., if it is not "-LO" in HALO.CNF). When a locator device exists, the default is that it is active. When active, some command options can be entered using the locator device. When not active, these options are entered using the keyboard for some of the commands (e.g. "VIEWPORT"), while some other commands (e.g. "BLOW", "TEXT") cannot be used.

4.1.3 DEBUG

This command is used for the debugging of DIDEC. A file name is prompted and useful information for debugging is written into this file. This command is left over from the original DIDEC development phase and is not needed under normal circumstances.

4.1.4 HELP

This gives information about a specific command, or lists the available commands. For information about a specific command, type

HELP xxx

where "xxx" is the name of the command. You can also type "HELP" without the command name, and then type the command name on the next line in response to the prompt. If you type "<CR>" in response to the prompt, you will get a list of the available commands.

4.1.5 HOST

This command temporarily suspends DIDEC and issues host (DOS) commands. If the user enters a DOS command, the command is executed and control is returned to DIDEC. If the user enters a blank line, control is returned to DIDEC. If the user enters the word "COMMAND", a sequence of DOS commands can be carried out. Enter "EXIT" to return control to DIDEC.

CAUTION: For systems with many things loaded in memory, the "HOST" command may lead to system failure. In this case a memory expander such as QEMM may have to be used to move some DOS-resident programs to high memory, thus saving more space for running DIDEC.

4.1.6 PRINT

This command sends the currently displayed screen image to the printer specified on the "-P" line in HALO.CNF. The image is also written to the file HALO.PIC. The program HALOREAD can be used to read any *.PIC file.

4.1.7 QUIT

This command exits DIDEC. If an output file is opened, "QUIT" gives a prompt to purge it before exiting. (See also the "STOP" command.)

4.1.8 ROUTE

This command allows entry of DIDEC commands from a file instead of from the keyboard. A file name is prompted and commands are read from the file, echoed on the terminal, and executed. When the end of the file is reached, command control is returned to the keyboard.

4.1.9 STOP

This command terminates DIDEC. If an output file is opened, it must be either CLOSEd or PURGEd before the "STOP" command can be used. (See also the "QUIT" command.)

4.2 DATA FILE HANDLING COMMANDS

The user can open up to 4 DIDEC (*.DID) files. These may be restricted to read-only (no modification allowed), and at most one can be opened as the current output (either with "OUTPUT" or "EDIT"). After being opened, all (input or output) files are referred to in terms of their reference file numbers, 1 to 4.

4.2.1 CLOSE

This command closes and saves an opened *.DID file. The file, file number, and any associated viewport are also freed. If the file is an output file, there is a prompt to compress it upon closing (default = no compression). An output file must be closed before terminating DIDEC, or it will be lost.

4.2.2 EDIT

This command opens and existing *.DID file as output for editing. Internally in DIDEC, a copy is done and the editing is actually done on the "invisible" file unit 5. The editing is only incorporated into the selected file when the output file is closed.

4.2.3 INPUT

This command opens an existing *.DID file for input (read-only).

4.2.4 LIST

This command lists all vertex coordinates, wire characteristics, and patch information of a *.DID file. The device or file to receive the listing is prompted.

4.2.5 OUTPUT

This command opens a new output *.DID file. DIDEC can only have one output file at a time. If a file with the same name already exists, a warning is issued (that the old file will be overwritten when the current output file is closed).

4.2.6 PURGE

This command closes a *.DID file and destroys it. If the file purged is the output file and was opened using "EDIT" then it is the edited copy that is purged, leaving the original intact.

4.2.7 RENAME

This command changes the name of the output file. If the output file was opened with "EDIT", the new name will come into effect when the file is closed.

4.2.8 TITLE

This command displays the title of a file (30 characters), which was entered when the *.DID file was first created to allow the storage of some

additional explanation of the file contents. For the output file there is an option of changing the title.

4.3 GRAPHIC DISPLAY MANIPULATION COMMANDS

To show the contents of a *.DID file on the graphics screen, use the "DISPLAY" command. The other commands in this group are all supplementary to "DISPLAY", for the selection of the many possible displaying configurations.

4.3.1 BLOW

This command zooms into part of a display window. A locator device is needed for this command, used to pick the lower-left and upper-right corners for the zoom-in area.

The "BLOW" command can be iterated. The "RESET" command displays the window in its pre-BLOW view.

4.3.2 CODE

This command selects the colour-coding field to be used for displays. The 3 fields for selection are (C)olor, (D)iameter, and (L)ength. The default coding (upon starting DIDEC) is by diameter.

The colour-coding table is selected by the "TABLE" command.

4.3.3 CYLINDER

This command enables or disables the drawing of wires as cylinders (with radius as specified in the radius entry of each wire). The normal state is disabled, when wires are drawn as line segments.

The cylindrical representation of wires is useful for NEC structures, where the "true" representation of the geometric file can be displayed.

4.3.4 DISPLAY

This command generates a display window, after the file number, viewport, and view are selected. The view can be any parallel projection identified by the azimuth and elevation angles. Three special projections are also provided: front, top, and side views which correspond to the views from the positive x, z, and y axis, respectively.

4.3.5 DOTS

This command enables or disables the drawing of vertices. If enabled (default), a small white dot is drawn to show each vertex. The display will be completed faster if "DOTS" is disabled.

4.3.6 FIELD

This command displays the "three arrows" (k,E,H) of an incident field. The parameters theta, phi, H_theta, and H_phi are as in EFIE.

4.3.7 IMAGE

This command writes the currently displayed screen image to a HALO *.PIC file. The program HALOREAD can be used to read any *.PIC file (See Section 2.4).

4.3.8 LABEL

This command needs the locator cursor. It displays a list of vertices (vertex ID number and coordinates) at/near the locator cursor position.

4.3.9 LOCATE

This command labels 3D locations on displayed structures. A label is any 1 or 2 character string and put at specified locations marked with a small "+".

The locations can be supplied through the keyboard, or read from a file. The first line of the file is an integer "n" giving the number of locations. Each of the subsequent n lines contain an "A2"-format label and the 3 real x,y,z coordinates of the location.

4.3.10 NUMBER

This command sets the vertex and tag displayed number size. The size number is an integer between 0 and 9. On starting DIDEC, the default size is 0 (no vertex and tag number displayed). The size numbers 1 to 8 correspond to the following standard "real" scaling factor of character size:

1	0.25
2	0.5
3	1.0
4	2.0
5	3.0
6	4.0
7	5.0
8	6.0

The size number 9 allows a choice of any positive real scaling factor. The size numbers for vertex and tag are independent and are set separately.

4.3.11 RESET

This command redraws the display to show the complete object contained in the file associated with the viewport. Reset is useful after "BLOW" and digitization ("SCALE", "DIGITIZE", and "CHANGE").

4.3.12 SPECTRUM

This command displays a current magnitude spectrum. This command is used with the "FROM CDDP" command, where the current magnitudes are scaled and stored in the COLOR field of each wire. One uses the "CODE = C" command to change the display colour to the COLOR field, then uses the "TABLE" command to select a COLOR table. Then the structure will be displayed in the current magnitude colours.

The "SPECTRUM" command selects the current magnitude scale stored in a *.CUR file and displays it on the bottom of a viewport. (Naturally, the same *.CUR used in the FROM CDDP scaling should be used.)

A *.CUR file has the format:

```
7     step 7 median value (positive real)
6     step 6 median value
5     step 5 median value
4     step 4 median value
3     step 3 median value
2     step 2 median value
1     step 1 median value
n
```

where n=0 for a linear scale and n=1 for a logarithmic scale.

4.3.13 TABLE

This command associates color-code, diameter, length, and patch area values with colour indices. There are 4 types of color-coding tables, for "colour", diameter, subdivision length, and patch area color-coding. (The colour field is used by CDDP files to display current magnitudes in colour, with the commands "CODE = C" and a chosen colour table.)

The tables are stored in separate files. Tables may be created, edited, or read in from files. To access a table, type the "TABLE" command and answer the questions asking for the type of table and the option to be used.

The read option prompts for a file name, reads the file contents, and exits the "TABLE" command. The edit option allows editing the table which is currently

in memory, and then prompts for a filename for the "new" table. The create option zeros the table arrays in memory and branches to the edit option. When a table is being edited (including the create option) the user is prompted for a table index from 1-15, a color number from 1-15 for color-coding, and a value as applicable for the diameter and subdivision length tables. If a locator device exists and the "ACTIVE" toggle is set then the color is selected using the cursor input method (if the table is displayed) instead of by color number. In this case, position the cursor on the leftmost of the 2 color bars.

4.3.14 TEXT

This command puts text at a locator cursor chosen position. Each text line may have up to 50 characters and there is a maximum of 10 text lines per viewport.

Available text characters are the 26 CAPITAL letters, the 10 numerals, the space character, and

$ * ( ) + - = , . / \ [ ] _ < > : ; |

All other entries will be displayed as the "unknown" character (three short horizontal lines); this is a limitation of DIGRAF.

4.3.15 VIEWPORT

This command chooses a display viewport configuration. One of the 9 configurations can be selected using the cursor (if active), or by entering "VIEWPORT i,j" where i,j=1,2,3.

4.4 MODEL BUILDING AND EDITING COMMANDS

The commands in this section deal with the actual geometric model creation process within DIDEC. Most of them may be classified into one of two groups, vertex manipulation commands and wire manipulation commands, although many commands that handle vertices also affect the attached wires, and many commands that handle wires also affect the incident vertices. Several commands, as well, do not fall into either category.

A vertex is identified by its number, a positive integer. Vertices are separated by commas, and can be specified as a range with a hyphen joining the initial and final vertices.

We shall first discuss three special vertex manipulation commands that use the digitizing tablet (specified in the "-T" line of HALO.CNF; see Section 2.2): "SCALE", "DIGITIZE", and "CHANGE".

4.4.1 DIGITIZATION

The digitization and construction of a 3D wire grid model from its 2D projections is done by first building a series of partial descriptions, called segments, which are subsequently merged to for the complete model. The reason for the segmentation is effectively twofold. First, the individual projects and sections for natural 2D segments which from then the 3D structure after merging. Second, the individual plans containing the 2D projections of the object to be modelled are usually much bigger in size than the digitization surface (tablet) available. Due to segmentation the scaling of the digitization surface is not as simple as it would be otherwise, in order to maintain correct dimensions and segment orientation/alignment throughout the entire model. Therefore, each time a new segment is to be digitized, the user must supply two vertices which lie on the digitization surface and whose real world coordinates must be made known to the system. For the first segment of a model, these coordinates must be entered manually, while for any subsequent ones the user may just give a reference using the Vertex Coordinate Specification (VCS) described previously, provided that the segments are partially overlapping at least in these two reference points.

Suppose now that the surface has been scaled for the given segment to be digitized. The user now may proceed in two relatively independent ways, namely, one can first specify the individual wires in terms of their endpoint Vertex Identifications (VIDs), or to proceed with the digitization of the vertex coordinates. If a wire is created whose VID is so far unknown to the system, an entry for this vertex is made in the database with the coordinate values left undefined. This approach is very useful, for it allows the user to specify the exact coordinates in one segment and use the vertex in many other segments for wire references. During the merging process, a defined coordinate from one segment supersedes any undefined values that may come from other segments. Moreover, when a display of a segment is requested, the system verifies whether all vertices have the coordinates required for the particular projected defined. Any deviations are reported to the user, in order that corrective action could be taken.

The vertex digitization is controlled using three commands: "SCALE", "DIGITIZE" and "CHANGE".

4.4.1.1 SCALE

This command sets the linear scale of the digitizing tablet using two vertices. This command must be called first, before the other two commands ("DIGITIZE" and "CHANGE") that use the digitizing tablet. The two vertices uses for scaling are digitized on the tablet, and their 3D coordinates are entered on the keyboard. These values are used for setting the scale.

In our sample configuration (Section 2.2), the vertex "n" is digitized on the CalComp 16-key cursor by positioning the crosshair over the vertex in the blueprint, and then pressing the number n (the digits in sequence) followed by "D" (for "done"). Mistakes can be erased by the key "C" (and then redoing the vertex). The key "A" aborts the command.

4.4.1.2 DIGITIZE

This command enters vertices of blueprints with a digitizing tablet. The vertex number of vertices to be digitized can either be entered at the "VERTICES?" prompt, or if none is entered here directly on the cursor keypad. In the latter case entering vertex "0" terminates this command. Commands "CHAIN" and "POLYGON" may be used to automatically create wires joining the new vertices.

4.4.1.3 CHANGE

This command re-digitizes existing vertices with a digitizing tablet. The vertex number of vertices to be re-digitized can either be entered at the "VERTICES?" prompt, or if none is entered here directly on the cursor keypad. In the latter case entering vertex "0" terminates this command.

4.4.2 VERTEX MANIPULATION

The following commands allow the manipulation of vertices.

4.4.2.1 ENTER

New vertices can also be created with this command. It accepts numerical keyboard entries of vertex coordinates to create new vertices.

4.4.2.2 DELETE

Existing vertices can be eliminated with this command. It deletes a vertex and all attached wires.

4.4.2.3 ADD

This command adds a constant to coordinates (X, Y, Z, or all 3) of chosen vertices of an output file.

4.4.2.4 DISCONNECT

This command finds all vertices from the output file with ≥ a supplied number of wires attached, and (optionally) deletes them. This command is used to make a *.DID file containing cells easier to view because of the huge number of edges. It also gives some indication as to which edges are on the "surface" of the structure and which are "interior".

4.4.2.5 MULTIPLY

This command multiplies coordinates (X, Y, Z, or all 3) of chosen vertices of an output file by a constant.

4.4.2.6 SOCK

Some codes, such as EFIE, do not allow edges to be shared by more than two surfaces. It is therefore necessary to "inflate" some surfaces before an input file can be created for these codes (e.g. "TO EFIE").

The "SOCK" command splits a vertex in 2 in the direction and distance entered, and doubles the connecting wires. An example is shown on Figure 1-4 where the right wing of an aircraft in shown before and after "inflation". The "EDGES" command (described in Section) can be used to find the "illegal" edges.

4.4.2.7 VERIFY

This command shows which vertices are fully or partially defined. This command is used after digitizing for checking. This command also resets the F*MIN and F*MAX variables. If the file is an output file, the new values will be written in the file header.

4.4.3 WIRE MANIPULATION

The following commands (primarily) manipulate wires. A wire is defined as a pair of vertex numbers (the two end points) separated by a blank. Wires are separated by commas.

4.4.3.1 BISECT

This command bisects chosen wires in an output file.

The process is:

1. cuts the old wire
2. create the mid-point vertex
3. join the two end-point vertices to the mid-point vertex.

4.4.3.2 CHAIN

This command creates a chain of wires by joining a list of vertices V1, V2,...,Vn. If Vn is to be joined to V1, completing a polygon, use the POLYGON command.

4.4.3.3 COLOR

This command allows either changing the colour of specified wires or setting the colour to be used for all wires which are created after this command is entered. First you must select the color index from 1 to 15. This is done using the cursor, if a locator exists, the "ACTIVE" tag is set, and the table is displayed. Position the cursor over the leftmost of the two color bars.

However, it is actually the index that you are selecting, so if the 2 color bars are different at that level then it is the rightmost color that you will see in future displays unless the table is changed.

4.4.3.4 CUT

This command removes wires.

4.4.3.5 DIAMETER

This command allows either changing the diameter of specified wires or setting the diameter to be used for all wires which are created after this command is entered. First you must select the color index from 1 to 15. This is done using the cursor, if a locator exists, the "ACTIVE" tag is set, and the table is displayed. Position the cursor over the leftmost of the two color bars. However, it is actually the index that you are selecting, so if the 2 color bars are different at that level then it is the rightmost color that you will see in future displays unless the table is changed.

4.4.3.6 EDGES

Some codes, such as EFIE, do not allow edges to be shared by more than two faces. Before creating an input file for these codes (e.g. "TO EFIE"), it is necessary to insure that this condition is not violated.

The "EDGES" command picks out all edges in a structure shared by more than two surfaces. The faces can be triangular, or triangular and quadrilateral. The quadrilateral option is very slow for large structures. See also the "SOCK" command, in Section 4.4.2.6.

4.4.3.7 JOIN

This command creates wires by joining a pair of vertices (the two end points) separated by a blank. Wires are separated by commas.

4.4.3.8 LENGTH

This command allows either changing the length of specified wires or setting the length to be used for all wires which are created after this command is entered. First you must select the color index from 1 to 15. This is done using the cursor, if a locator exists, the "ACTIVE" tag is set, and the table is displayed. Position the cursor over the leftmost of the two color bars. However, it is actually the index that you are selecting, so if the 2 color bars are different at that level then it is the rightmost color that you will see in future displays unless the table is changed.

4.4.3.9 POLYGON

The POLYGON command creates a chain of wires by joining a list of vertices V1, V2,...,Vn, and completing the chain into a polygon by joining Vn to V1. If Vn is not to be joined to V1, use the "CHAIN" command.

4.4.3.10 RADIUS

This command changes the absolute radius value (positive real number) of a wire, either by direct entry of a new radius or by entry of a multiplying factor for the old radius.

4.4.3.11 SEGLEN

This command changes the absolute segment length (positive real number) of a wire. Note segment length (in the NEC context) is not wire length: a wire can have several segments.

4.4.3.12 TAG

This command gives a wire a NEC-compatible identifiable tag. All new wires created in DIDEC (by "JOIN", "CHAIN", ...) have tag = 0 and must be re-tagged.

4.4.3.13 TRIANG

This command splits each quadrilateral patches into two triangular patches. It must used before creating input files from a model containing quadrilateral patches (e.g. from a *.DID file "FROM [NEC]") for codes such as EFIE which uses only triangular patches. This command is very slow for large structures.

4.4.4 MISCELLANEOUS

4.4.4.1 PATCH

This command creates triangular and quadrilateral patches. These patches are NEC-compatible and will be converted to NEC patches by the "TO [NEC]" command. The colour of each patch reflects its area, coded according to the current area table. The patching can be done manually (by selecting the patch vertices with the locator device) or automatically (creating patches from all triangular or quadrilateral wire frames). Automatic quadrilateral patching of large structures is very slow.

4.4.4.2 MERGE

This command merges an input file into the output file allowing the concatenation of several *.DID files. Complicated structure can be easily created from simple pieces or submodels. The common vertices (i.e. those at the same locations) from different input files must have the same vertex numbers to merge properly.

4.4.4.3 REFLECT

This command reflects a file along a principal axis (x,y,z) into the output file. The coordinate of each vertex corresponding to the axis of reflection is negated. The original vertices must all lie on the same side of the axis. Vertices with a zero entry in the reflected coordinate will get duplicated. So additional vertex and wire manipulations may be required to "tidy up". The vertex numbers are changed by adding 1000 for x-axis reflection, 2000 for y-, and 4000 for z-.

4.5 DATA FILE CONVERSION COMMANDS

The prime purpose for the development of the DIDEC program is the creation of geometric input files for the many numerical electromagnetic simulation codes. A DIDEC file can be generated by conversion from the geometry file of one of these codes, and a DIDEC file can be converted to yield the geometry file of one of these codes.

4.5.1 FROM

This command creates a DIDEC output *.DID file from a geometric input file of one of the electromagnetic simulations programs.

4.5.1.1 FROM CDDP

The CDDP *.STR file is a NEC input file appended with the current magnitudes on each wire segment for each frequency. An *.STR file can be generated by the independent program NECDDP, which reads from a NEC output file the currents on the segments and appends them to a NEC input file. The current magnitudes are translated into a colour scale and stored in the colour field for CODE = C display. The current magnitudes are scaled in 7 steps either linearly by maximum or by an existing *.CUR current scale file. In the former case the scale used will be written to a *.CUR file (for use by the SPECTRUM command for display).

A *.CUR file has the format:

```
7     step 7 median value (positive real)
6     step 6 median value
5     step 5 median value
4     step 4 median value
3     step 3 median value
2     step 2 median value
1     step 1 median value
n
```

where n=0 for a linear scale and n=1 for a logarithmic scale.

4.5.1.2 FROM EFIE

Although EFIE is a patch-only code, it requires that the geometry be entered as a list of vertices (the corners of the patches) and a list of the edges forming triangular patches (it does not support quadrilateral patches). Therefore, when a EFIE input file (*.EFI) is read in, it is entered as a wire grid model. The "PATCH" command can be used to convert the model into a patch model. Since the EFIE input file contains no radius information (the line segments joining vertices are "edges" and not "wires"), in the conversion the diameter of each "wire" of the DIDEC file is assumed to be 10% of the wire length. The 10% value is chosen so that the "equal-area" rule of wire-grid modelling (i.e., the cylindrical walls of the wires have the same area as the grids) is approximately satisfied. The radius of specific wires can later be changed with the "RADIUS" command.

4.5.1.3 FROM FDTD

This command converts a *.FDG file of the finite difference time domain code, FDTD from the University of Manitoba, to DIDEC format for display. An *.FDG file has the format:

```
nx  ny  nz
number_of_occupied_cells
   I   J   K
       :
       :
"DX = " dx  "DY = " dy  "DZ = " dz
"RANGE OF  I = [..,..]  J = [..,..]  K = [..,..]"
```

(See the documentation of FDTD for details [6]).

4.5.1.4 FROM NEC

The conversion routine can only handle patches (SP, SC, and SM cards) and wires (GW cards). All other geometric specifications (symmetry GX, etc.) must first be explicitly rewritten into the acceptable format in the NEC input file *.NCI prior to the DIDEC conversion.

4.5.1.5 FROM THREDE

This command converts a *.TDG geometry file of the finite difference time domain program THREDE into a DIDEC file. The purpose is simply to display the cellular structure.

A *.TDG file has the format:

```
"TITLE"
"X0"   nx
   X0(1)
     :
   X0(nx)
"Y0"   ny
   Y0(1)
     :
   Y0(ny)
"Z0"   nz
   Z0(1)
     :
   Z0(nz)
"I,J,K,NOPE(I,J,K)"
   i   j   k   NOPE
       :
       :
```

(See the documentation of THREDE for details [7]).

The *.TDG file is closely related to the geometry file *.FDG. An *.FDG file can easily be converted to a *.TDG file (and vice versa). There is a major difference between THREDE and FDTD geometry: the (i,j,k) system of THREDE is right-handed with the orientation i = aft, j = top, and k = portside; while the (I,J,K) system of FDTD is identical to that of DIDEC, right-handed with I = front, J = portside, and K = top. The conversion is I = nx-i, J = k, and K = j.

4.5.1.6 FROM TWTDA

The TWTDA input file *.TWI used in this conversion is the version 3 developed at DREO, and has only 1 comment line and the DT... line before the geometric data.

4.5.2 TO

This command generates a geometric input file for the chosen electromagnetic simulations code. The DIDEC file used to obtain the data can be either an input file or an output file of the simulation code.

4.5.2.1 TO EFIE

In the conversion from a *.DID file to an *.EFI file, diameter and subdivision length tables are ignored (because EFIE does not use these).

Additional considerations before the conversion are:

1. that all "grids" must be triangular (because EFIE works on triangular faces). The DIDEC command TRIANG can be used first to convert each quadrilateral grid to two triangular pieces.

2. that EFIE has a geometric restriction that an edge cannot be shared by more than two triangles. The DIDEC command EDGES can be used to identify all the edges that are shared by more than two grids, and the DIDEC command SOCK can be used to "inflate" the extraneous "wings" into thin tubes.

4.5.2.2 TO FDTD

The *.DID file to be converted must consist of triangles only. The minimum cell edge size provided is such that the overall cell space is at most $100\chi100\chi100$ (hence the object can at most occupy the central $50\chi50\chi50$ cells).

The default for the cell selection criterion is the equivolume sphere (a sphere with a radius of $0.62035 = 0.5\cdot(6/\pi)^{1/3}$ which has a volume of 1.), which is usually adequate.

The file generated by this command is a *.FDG file in the format described in Section 4.5.1.3. The "FROM FDTD" command can be used to convert this *.FDG file for display. Note that the "FROM FDTD" and "TO FDTD" commands are not operational inverses.

4.5.2.3 TO NEC

If you did not enter coordinates in either meters or feet, you must enter the number of meters in each unit. For example, if using inches you could type "1/39.25" (without the quotes). Remember that an expression can be typed, you do not need to reach for a calculator.

In the prompts for obtaining radius and segment length from the lookup tables, the default Y uses the diameter and segment length tables, while the response N uses the actual diameter and segment length entries of each wire.

4.5.2.4 TO THREDE

The *.DID file to be converted must consist of triangles only. The minimum cell edge size provided is such that the overall cell space is at most $100\chi100\chi100$ (hence the object can at most occupy the central $50\chi50\chi50$ cells).

The default for the cell selection criterion is the equivolume sphere (a sphere with a radius of $0.62035 = 0.5 \cdot (6/\pi)^{1/3}$ which has a volume of 1.), which is usually adequate.

The file generated by "TO THREDE" is a *.TDG file which contains the XO, YO, and ZO arrays and the I,J,K,NOPE for each cell (each NOPE=4). To see what this *.TDG file looks like, use "FROM THREDE" to convert it back into DID format. Note that "TO THREDE" and "FROM THREDE" are NOT operational inverses: the resulting *.DID after TO and FROM is the "THREDE equivalent" of the original *.DID file.

4.5.2.5 TO TWTDA

In the prompts for obtaining radius and segment length from the lookup tables, the default Y uses the diameter and segment length tables, while the response N uses the actual diameter and segment length entries of each wire.

The *.TWI file generated has only 2 non-geometric lines in the beginning; this is version 3 of TWTDA developed at DREO.

5.0 ASSOCIATED PROGRAMS

Several programs associated with DIDEC have been developed at DREO. HEDRON creates simple polyhedral pieces for merging in DIDEC to form complicated geometric structures. The programs NECDDP, EFCDDP and TWCDDP extract current magnitude from the output files of NEC, EFIE and TWTD respectively, and create the corresponding *.STR files in CDDP format for the DIDEC "FROM CDDP" command.

5.1 HEDRON

HEDRON is a FORTRAN program, developed at DREO, that creates polyhedra in the following simple geometric shapes:

1. box
2. cylinder
3. cone
4. sphere
5. rectangular plate
6. prism
7. polygon
8. torus
9. paraboloid of revolution

The running of the program is entirely interactive. The user is prompted for the dimensions of the object, the maximum edge length, the option to make only part of the polyhedron, triangulation, rigid transformations, and the choice of initial vertex and edge numbering.

The polyhedron is built as an *.EFI file (i.e. the output of running HEDRON in the form

HEDRON filename

is the EFIE input file filename.EFI). The *.EFI file may then be converted by DIDEC into a *.DID file for further processing. Note that if the triangulation option is not chosen, then the *.EFI file contains quadrilateral grids and is therefore not a proper EFIE input (EFIE takes only triangles). The *.EFI format is chosen for its mathematical convenience in polyhedral representation. If an *.NCI file is desired, for example, the DIDEC commands FROM EFIE followed by TO NEC would convert the *.EFI file to *.NCI.

The principal idea behind HEDRON is that simple geometric shapes can be created easily. More complicated objects may then be built by the "MERGE" command, using several simple geometric parts. Other wire and vertex manipulations (such as opening up "windows") may also be carried out in DIDEC. The finished product may then be converted in turn to the *.EFI, *.FDG, *.NCI, *.TWI, or *.TDG format. Figure 5-1 shown an example of an object built with this method (from cylindrical and conical segments).

The "maximum length for edges" choice prompts the user to enter a positive real number. Note however that this number is the maximum for ALL edges generated, including, in particular, the "diagonals" of quadrilateral grids.

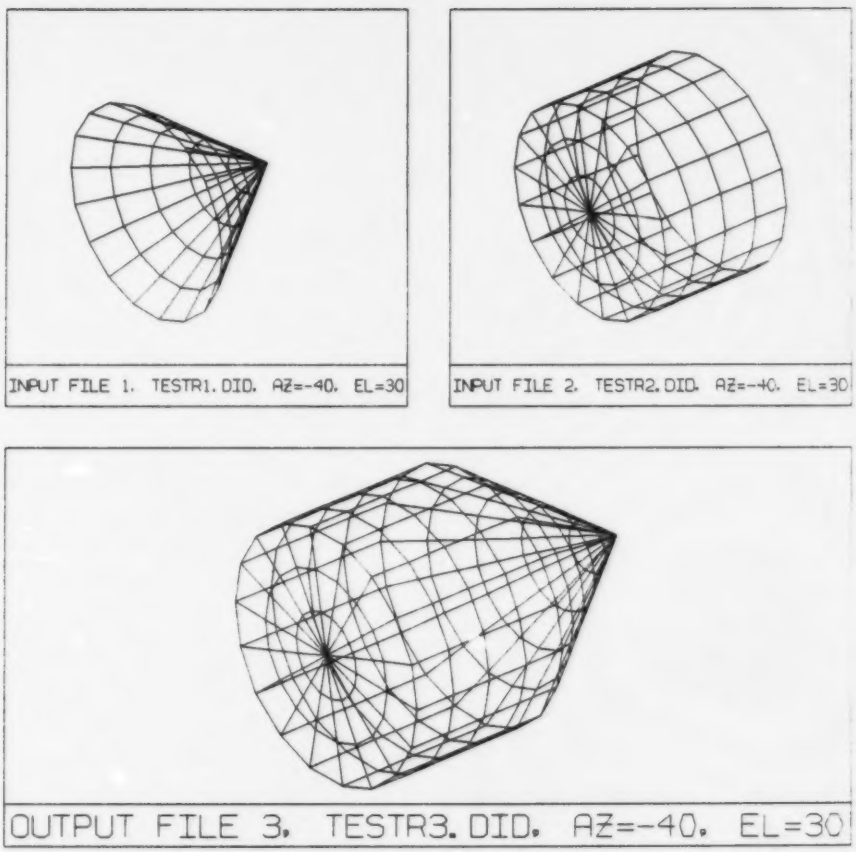

Figure 5-1 Example of an object created by merging and editing simple shapes generated by HEDRON.

Thus, for example, if one wants the edges of the square grids to be at most 1 unit long, then the maximum length for edges should be entered as a number slightly larger than 1.4142 = SQRT(2).

In general the geometric object is positioned so that it is "centered" at the origin and with the z-axis as the principal axis. One may choose any number of sequential rigid transformations (translations and rotations about a cartesian axis) to reposition the object. The transformations are important in merging common vertices. As well, in the MERGE command of DIDEC common vertices from different files (i.e. vertices from different objects that occupy the same locations) must have identical numbers (VIDs) to merge properly. (If vertices have different VIDs they will be treated as different vertices even if they have the same coordinates.) This is what the option to choose the numbering of the initial vertex is for. (The numbering of the initial edge is not important and can be chosen as "1".)

For most of the geometric shapes the grids created are quadrilateral, unless the triangulation option is chosen, in which case "diagonals" are added to each grid. For the other shapes the grids are triangular. In the following we shall discuss each of the simple geometric shapes in turn.

5.1.1 BOX

A box is six rectangular faces joined at right angles. The user is prompted to give first the overall (x,y,z) dimensions, and then the maximum edge length. Then the option to triangulate is given, followed by the choice of transformations and initial vertex and edge numbering.

Before the choice of transformations, a topological summary of the geometric object is given, in terms of the number of vertices V, edges E, and faces F, and the Euler characteristic X = V-E+F. The box built by HEDRON is a simply connected surface (topologically equivalent to the sphere), hence X = 2.

5.1.2 CYLINDER

A cylinder is a geometric object with circular transverse sections (i.e. a surface of revolution). HEDRON actually generates a polyhedral approximation of a cylinder, thus the transverse sections are in fact regular polygons.

One can choose to create a right circular cylinder, in which case the radius and height h are prompted. The z-axis is the central axis of symmetry and the range of z is from -h/2 to h/2.

An alternative is to generate a cylinder from a "wall file" *.GCN by revolving around the z-axis. The *.GCN file has the following structure:

```
n = number of generator points
x(1)  z(1)
      :
      :
x(n)  z(n)
m = number of divisions of the circumference
```

(so there are n-1 edges along the "wall" and the transverse sections are regular m-gons).

The edge maximum, triangulation, etc., options are as in the box. With the cylinder one has the choice of whether to build the top and bottom disks, thus resulting in a cylinder closed in both ends, a cylinder with one open and one closed end, or a cylinder with two open ends. The Euler characteristic X of the three cases are 2, 1, and 0, respectively.

5.1.3 CONE

With the cone there is a choice of generating a right circular cone (a surface of revolution about the z-axis with a right triangle as generator) or a frustum (a cone with the "point" removed).

For the right circular cone the radius of the cone base and the height h have to be supplied. Then the other options are as before. There is an additional option whether to build the base disk, with "yes" yielding X = 2 and "no" yielding X = 1. In both cases the "point" is in the +z direction and the z range is from -h/2 to h/2 (unless transformed).

With the frustum the radii of the top and bottom disks and the height are prompted. The other options are as in the cylinder (including whether to build the top and bottom disks). An additional option is the "shearing". With no shearing, on gets a right circular frustum which is identical to a cylinder generated by a straight line segment; the central axis of symmetry is the z-axis. With shearing to the front, the "back" of the frustum is aligned with the z-axis (hence perpendicular to the top and bottom disks); with shearing to the back, the "front" of the frustum is aligned with the z-axis. The shearing option is very useful in building nose and tail cones of aircrafts, where the transverse section are circular but the structure is "sheared" (with no axis of rotational symmetry). For example, the "tail" of the helicopter EH101 in Figure 5-2 is a sheared frustum.

5.1.4 SPHERE

The HEDRON version of a sphere is in fact a "geodesic dome". Only the spherical radius has to be provided. An extra option is whether to force the equator to be a grid line (i.e. an even division of a longitude). The Euler characteristic X of the sphere is, of course, 2.

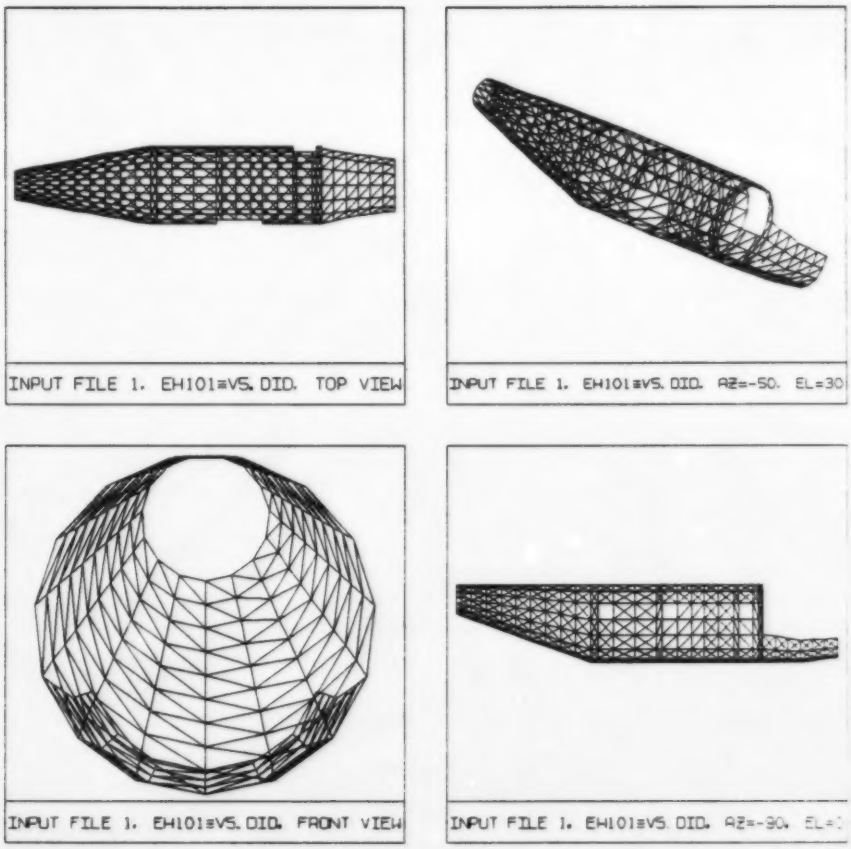
INPUT FILE 1. EH101=VS. DID. TOP VIEW
INPUT FILE 1. EH101=VS. DID. AZ=-50. EL=30
INPUT FILE 1. EH101=VS. DID. FRONT VIEW
INPUT FILE 1. EH101=VS. DID. AZ=-90. EL=

Figure 5-2 Multiple views of an object (the EH101 helicopter) generated by HEDRON. The tail section is a sheared cone (frustum).

5.1.5 RECTANGULAR PLATE

A rectangular plate is in effect one face of a box. After conversion to DIDEC, a rectangular plate can be easily converted into any quadrilateral shape by moving the vertices with the commands "ADD" and "MULTIPLY".

5.1.6 PRISM

A prism is a normal translation of a polygon (i.e. a polyhedron with two identical polygonal ends and with the corresponding polygonal vertices joined by line segments perpendicular to the plane of the polygons).

The polygon is provided in a generator file *.GEN in the following format:

```
n
x(1)   y(1)
     :
     :
x(|n|) y(|n|)
```

where |n| is the number of points in the generating polygon (with n<0 meaning that the polygon is closed, i.e. the |n|th vertex joining back to the first vertex, and with n>0 meaning that the polygon is open). The height h of the prism is then prompted. The generating polygon is parallel to the (x,y)-plane and is translated in the z direction from z=-h/2 to h/2.

The walls of the prism are divided up either according to the supplied edge maximum, or according to an external division file, in which case only the walls of the prism are generated without filling in the top and bottom polygons. (The polygons may be "filled" with the POLYGON option, discussed below.) The wall division file *.WAL has the format:

```
m
r(1)
r(2)
 :
r(m)
```

where m is the number of values in the file, and (r(i)) is an increasing sequence between 0 and 1 that divides the height h into m+1 parts. For example, if h=5.0 and the *.WAL file is

```
3
0.19
0.46
0.73
```

then the vertices on the walls would have z coordinates of $2.5\cdot(h/2)$, $1.55\cdot(h/2 - 0.19\cdot h)$, $0.2\cdot(h/2 - 0.46\cdot h)$, $-1.15\cdot(h/2 - 0.73\cdot h)$, and $-2.5\cdot(-h/2)$.

5.1.7 POLYGON

A polygon is generated in HEDRON by a file *.GEN in the same format as the generator in PRISM. The file *.GEN provides the "perimeter" of the polygon. The grids are triangular. The resulting filled polygon (before any transformations) lies on the (x,y)-plane. There is a choice of filling in the polygon "arithmetically" or "geometrically". In both cases the number of divisions of a radius (i.e. line segment joining the centre of the polygon to a polygonal vertex) is prompted.

In the "arithmetic" filling case, the number of divisions must be a divisor of the number of points in the generator file. Going in from the perimeter to the centre, the number of triangles decreases arithmetically.

In the "geometric" filling case, the number of divisions n must be such that 2^{n-1} divides the number of points in the generator file. Going in from the perimeter to the centre, the number of triangles decreases geometrically (each subsequent inner "layer" has half as many triangles).

5.1.8 TORUS

For the torus, the inner and outer radii have to be provided. The edge maximum, triangulation, and transformation options follow. The Euler characteristic of the torus (genius = 1) is 0. The non-transformed torus has centre at the origin and lies "on" the (x,y)-plane.

5.1.9 PARABOLOID

The paraboloid is a surface of revolution of an (x,z)-plane parabola about the z-axis (thus the apex is at the origin and the axis of rotational symmetry is the z-axis), before any transformations. The focal length and the (rim) radius of the paraboloid are to be provided. In addition to the usual options, one can choose to number the vertices outwards or inwards, i.e. to choose the apical vertex as the initial or final vertex. The paraboloid is useful in modelling dish antennae.

5.2 ??CDDP

CDDP is a FORTRAN program on the VAX that uses the graphics library PLOT10. It displays the current distribution of a wire grid structure as dashed line segments of lengths proportional to current magnitude, at the centres of the wire segments.

In DIDEC, there is the option of displaying the current distribution by colour: the current magnitude of each wire segment is assigned a colour that is graded on a specified colour scale. (See the "FROM CDDP" command in Section 4.5.1.1, and the CODE, TABLE, and SPECTRUM commands in Section 4.3.) For simplicity the same input file format used in the program CDDP is used in the "FROM CDDP" command of DIDEC.

The CDDP input file has the default extension *.STR, and is a NEC input file *.NCI appended with the current magnitudes on each wire segment for each frequency. That is, it looks like:

```
CM .....
CE
GW .....
GW .....
  :
GW .....
GE
FR .....
  :
EN
FREQUENCY  freq_1 MHZ
CURRENTS
curr_mag  curr_phase
      :
curr_mag  curr_phase
FREQUENCY  freq_2 MHZ
CURRENTS
curr_mag  curr_phase
      :
```

To generate such a file *.STR, several small programs are available.

The program NECDDP extracts the currents on each wire segment for each frequency from NEC files. First, the NEC input file *.NCI is copied to *.STR. Then NECDDP is run with the file name prefix, e.g. TEST, then currents from TEST.NCO (obtained from running NEC with TEST.NCI) are read and then appended to the file TEST.STR. Interactive options include the choice of scaling the currents by the wire radii (i.e. in the unit A/m = "J/Hinc"), or just in the (unchanged) unit Amperes. The former scaling is useful in comparing NEC with EFIE results because EFIE currents are in J/Hinc. There is also the option of appending only the currents of one chosen frequency to *.STR, or writing all the frequencies in the *.NCO file.

The program EFCDDP extracts the currents from an EFIE output file *.EFO. But since we are using the standard format *.STR, the process here is slightly more complicated. First, the EFIE input file, e.g. TEST.EFI, is converted by DIDEC to a NEC input file, e.g. TEST.NCI (with the commands "FROM EFIE" and "TO NEC"). Then TEST.NCI is copied (or renamed) to TEST.STR. Then EFCDDP is run with the command line option TEST, and currents from TEST.EFO are read and appended to TEST.STR. Here the currents are in the unit A/m (J/Hinc) and are for the one frequency that was used when TEST.EFO was generated by running EFIE.

The program TWCDDP extracts the maximum current (in time) from a TWTDA output file. Again, the TWTDA input file TEST.TWI has to be converted to a NEC input file TEST.NCI using DIDEC ("FROM TWTDA" + "TO NEC") first, and then TEST.NCI is copied/renamed to TEST.STR. Running TWCDDP TEST results in the maximum current on each wire segment appended to TEST.STR. (TWTDA is a time-domain code, so we are setting the "frequency" zero and picking the maximum

currents as the current distribution indicator.) There is the choice of the current unit in A or A/m.

INDEX

ABORT command 20
ACTIVE command 20, 26, 29, 30
ADD command 28, 42
BISECT command 29
BLOW command 23, 25
Box, generating a 39
Cell models 5, 13
CHAIN command 28, 29, 31
CHANGE command 25, 27, 28
CLOSE command 21
CODE command 23, 43
COLOR command 13, 29
Command syntax 16
Cone, generating a 40
Configuration 9
  graphics tablet 10
  HALO.CNF file 9
  mouse 10
  printer 10
CUT command 30
CYLINDER command 23
Cylinder, generating a 39
DEBUG command 20
DELETE command 28
DIAMETER command 14, 30
DIGITIZE command 15, 25, 27
DISCONNECT command 28
DISPLAY command 14, 23
DOTS command 23
EDGES command 14, 29, 30
EDIT command 22
EFCDDP program 44
EFIE 1, 3, 5, 13, 14, 29-31, 33, 35, 37, 44
ENTER command 28
FDTD 1, 3, 5, 13, 15, 33-35
FIELD command 24
FROM CDDP command 25, 32, 37, 43
FROM EFIE command 33, 44
FROM FDTD command 15, 33, 35
FROM NEC command 33
FROM THREDE command 15, 33, 36
FROM TWTDA command 34, 44
Graphics tablet 10
HALO 9, 10
HALO.CNF configuration file 9
HALOREAD program 12
HEDRON program 37
HELP command 18, 20
HOST command 20
IMAGE command 24
INPUT command 22
Installation 9
JOIN command 30
LABEL command 24
LENGTH command 30
LIST command 22
LOCATE command 24
MERGE command 31, 37
Mouse 10
MULTIPLY command 28, 42
NEC 1, 3, 5, 13, 14, 31-33, 35, 37, 44
NECDDP program 44
NUMBER command 24
OUTPUT command 22
Paraboloid, generating a 43
PATCH command 14, 31, 33
Patch models 1, 3, 13, 14
Patches 1, 3, 5, 13, 14, 22, 25, 31, 33
Plate, generating a recangular 40
POLYGON 30
POLYGON command 28
Polygon, generating a 42
PRINT command 21
Printer 10
Prism, generating a 42
PURGE command 22
Quadrilateral patches 3
QUIT command 21
RADIUS command 31
REFLECT command 32
RENAME command 22
RESET command 23, 25
ROUTE command 21
SCALE command 15, 25, 27
SEGLEN command 31
SOCK command 5, 14, 29, 30
SPECTRUM command 25, 43
Sphere, generating a 40
STOP command 21
TABLE command 14, 23, 25, 43
TAG command 31
TEXT command 26
THREDE 1, 3, 5, 13, 15, 34, 35
TITLE command 22
TO EFIE command 14, 34
TO FDTD command 15, 35
TO NEC command 14, 31, 35, 44
TO THREDE command 15, 35
TO TWTDA command 36

Torus, generating a 43
TRIANG command 14, 31
Triangular patches 3
TWCDDP program 44
TWTD 1
TWTDA 3, 13, 34, 36, 37, 44
VCS (see also Vertices) 17
VERIFY command 29
Vertices 1, 13, 15-17, 22-24, 26,
27, 28, 29, 32
VID 27, 39
VIEWPORT command 15, 26
Viewports 15
Wire grid models 1, 3, 13
Wires 1, 3, 13, 14, 16, 22, 25,
27, 28-31, 43, 44
Zoom 23

REFERENCES

[1] G. L. Burke and A. J. Poggio, "*Numerical Electromagnetic Code (NEC) - Method of Moments*", Naval Ocean Systems Centre, Technical Document 116, July 1977.

[2] S. M. Rao, D. R. Wilton, and A. W. Glisson, "*Electromagnetic Scattering by Surfaces of Arbitrary Shape*", IEEE Transactions on Antennas and Propagation, Vol. AP-30, No. 3, May 1982, pp. 409-418.

[3] J. A. Landt, E. K. Miller, and M. Van Blaricum, "*WT-MBA/LLL1B: A Computer Program for the Time-Domain Electromagnetic Response of Thin-Wire Structures*", Lawrence Livermore Laboratory, May 1974.

[4] K. Umashankar and A. Taflove, "*A Novel Method to Analyze Electromagnetic Scattering of Complex Objects*", IEEE Transactions on Electromagnetic Compatibility, Vol. EMC-24, No. 4, 1982, pp. 397-405.

[5] D. Gaudine and S. J. Kubina, "*DIDEC Commands/ Structure/Installation*", Report No. TN-EMC-86-05, Concordia University, August 29, 1986.

[6] W.A. Chamma and L. Shafai, "*EMP Response of Arbitrary Conducting and Complex Objects*", Report for DSS contract W7714-8-5648/01-SS, University of Manitoba, December 1989.

[7] K.S. Kunz and B.W. Torres, "*THREDE User's Manual*", Report for contract N60921-77-C-0117, Mission Research Corporation, December 1977.

UNCLASSIFIED

SECURITY CLASSIFICATION OF FORM
(highest classification of Title, Abstract, Keywords)

DOCUMENT CONTROL DATA

(Security classification of title, body of abstract and indexing annotation must be entered when the overall document is classified)

1. **ORIGINATOR** (the name and address of the organization preparing the document. Organizations for whom the document was prepared, e.g. Establishment sponsoring a contractor's report, or tasking agency, are entered in section 8.)
Defence Research Establishment Ottawa
Ottawa, Ontario
K1A 0Z4

2. **SECURITY CLASSIFICATION** (overall security classification of the document including special warning terms if applicable)
UNCLASSIFIED

3. **TITLE** (the complete document title as indicated on the title page. Its classification should be indicated by the appropriate abbreviation (S,C or U) in parentheses after the title.)
A SOPHISTICATED CAD TOOL FOR THE CREATION OF COMPLEX MODELS FOR ELECTROMAGNETIC INTERACTION CODES (U)

4. **AUTHORS** (Last name, first name, middle initial)
DION, Marc; KASHYAP, Satish; LOUIE, Aloisius

5. **DATE OF PUBLICATION** (month and year of publication of document)
JUNE 1991

6a. **NO. OF PAGES** (total containing information. Include Annexes, Appendices, etc.)
57

6b. **NO. OF REFS** (total cited in document)
7

7. **DESCRIPTIVE NOTES** (the category of the document, e.g. technical report, technical note or memorandum. If appropriate, enter the type of report, e.g. interim, progress, summary, annual or final. Give the inclusive dates when a specific reporting period is covered.
DREO Technical Note

8. **SPONSORING ACTIVITY** (the name of the department project office or laboratory sponsoring the research and development. Include the address.
NUCLEAR EFFECT SECTION, Electronics Division Defence Research Establishement Ottawa
Ottawa, Ontario, K1A 0Z4

9a. **PROJECT OR GRANT NO.** (if appropriate, the applicable research and development project or grant number under which the document was written. Please specify whether project or grant)
Project 041LT

9b. **CONTRACT NO.** (if appropriate, the applicable number under which the document was written)

10a. **ORIGINATOR'S DOCUMENT NUMBER** (the official document number by which the document is identified by the originating activity. This number must be unique to this document.)
DREO TECHNICAL NOTE 91-16

10b. **OTHER DOCUMENT NOS.** (Any other numbers which may be assigned this document either by the originator or by the sponsor)

11. **DOCUMENT AVAILABILITY** (any limitations on further dissemination of the document, other than those imposed by security classification)

(X) Unlimited distribution
( ) Distribution limited to defence departments and defence contractors; further distribution only as approved
( ) Distribution limited to defence departments and Canadian defence contractors; further distribution only as approved
( ) Distribution limited to government departments and agencies; further distribution only as approved
( ) Distribution limited to defence departments; further distribution only as approved
( ) Other (please specify):

12. **DOCUMENT ANNOUNCEMENT** (any limitation to the bibliographic announcement of this document. This will normally correspond to the Document Availability (11). however, where further distribution (beyond the audience specified in 11) is possible, a wider announcement audience may be selected.)
Unlimited Announcement

UNCLASSIFIED

SECURITY CLASSIFICATION OF FORM

UNCLASSIFIED

SECURITY CLASSIFICATION OF FORM

13. **ABSTRACT** (a brief and factual summary of the document. It may also appear elsewhere in the body of the document itself. It is highly desirable that the abstract of classified documents be unclassified. Each paragraph of the abstract shall begin with an indication of the security classification of the information in the paragraph (unless the document itself is unclassified) represented as (S), (C), or (U). It is not necessary to include here abstracts in both official languages unless the text is bilingual).

(U) This report describes the essential features of the MS-DOS version of DIDEC_DREO, an interactive program for creating wire grid, surface patch, and cell models of complex structures for electromagnetic interaction analysis. It uses the device-independent graphics library DIGRAF and the graphics kernel system HALO, and can be executed on systems with various graphics devices.

(U) Complicated structures can be created by direct alphanumeric keyboard entry, digitization of blueprints, conversion from existing geometric structure files, and merging of simple geometric shapes. A completed DIDEC geometric file may then be converted to the format required for input to a variety of time domain and frequency domain electromagnetic interaction codes.

(U) This report gives a detailed description of the program DIDEC_DREO, its installation, and its theoretical background. Each available interactive command is described. The associated program HEDRON which generates simple geometric shapes, and other programs that extract the current amplitude data from electromagnetic interaction code outputs, are also discussed.

14. **KEYWORDS, DESCRIPTORS or IDENTIFIERS** (technically meaningful terms or short phrases that characterize a document and could be helpful in cataloguing the document. They should be selected so that no security classification is required. Identifiers, such as equipment model designation, trade name, military project code name, geographic location may also be included. If possible keywords should be selected from a published thesaurus. e.g. Thesaurus of Engineering and Scientific Terms (TEST) and that thesaurus-identified. If it is not possible to select indexing terms which are Unclassified, the classification of each should be indicated as with the title.)

Computer-Aided-Design
DIDEC
Digitize, Display and Edit code
Electromagnetic Interaction Analysis
Modelling
Cell modelling
Patch modelling
Wire grid modelling
FDTD
Finite Difference Time Domain code
EFIE
Electric Field Integral Equation code
NEC
Numerical Electromagnetic Code
THREDE
Three-dimensional Finite Difference code
TWTD